Dungeon

112550013 周廷威

April 22, 2024

Implementation detailed

Start System

玩家在遊戲開始前會有啟動的畫面，如下所示：

```
________
\______ \  __ __  ____    ____   ____  ____   ____
 |    |  \|  |  \/    \  / ___\_/ __ \/  _ \ /    \
 |    `   \  |  /   |  \/ /_/  >  ___(  <_> )   |  \
/_______  /____/|___|  /\___  / \___  >____/|___|  /\
        \/           \//_____/      \/           \/

Please enter your name:
```

此 ASCII Art 是以 figlet 產生的，並在輸入玩家名稱後，會有簡單的遊戲介紹：

```
Player userwei, welcome to the Dungeon Game!
Here are some survival rules:
1. There are four types of rooms: Normal, Desert, Forest, and Swamp. Each type has unique debuffs, items, NPCs, and more.
2. You can move up, down, left, or right. If you reach the exit room, you win the game.
3. You can interact with objects in the room, including monsters, NPCs, and items.
4. You can check your status, equipment, food, and manage game saves.
5. Eating food restores your hunger and thirst levels.
6. Equip items to increase your attack and defense stats.
7. You can save your game progress and load it later to continue playing.
8. If you die, you lose the game.

Press any key to start the game...
```

在按下任意鍵後，遊戲開始，並進入 StartRoom。

Actions Menu

玩家在進入每個房間時，都可以選擇自己的行動，如下所示：

```
------------------- ROOM -------------------
> Room type: normal
> Hunger consumption: 0
> Thirst consumption: 0
0: NPC Guide Ko
1: sword Wood Sword
--------------------------------------------
(M)ove
(S)how Status
(E)quipment
(F)ood
(C)reate Record
(L)oad Record
(Q)uit
Enter the instruction or the index of the charcter you want to interact: >
```

這部分的功能是在 Dungeon.cpp 的 chooseAction() 中實現的，並且在選擇完行動後，會進入相對應的函數，如：handleMovement()、player.triggerEvent(&player)、player.equip() 等。此函數在 RunDungeon() 中的 while 迴圈中被呼叫，直到抵達出口或 checkGameLogic() 回傳 false。

Movement

在上述的 Actions Menu 中，選擇移動後，會進入 handleMovement() 函數，如下所示：

```
(D)own Room
(L)eft Room
(R)ight Room
Which room do you want to go? >
```

此為非遊戲出口的房間，玩家可選擇可供移動的房間，地圖如 Room System Design 所示，並且在選擇後，進入相對應的房間。
若為出口的房間，則會有 (E)xit 的選項：

```
(R)ight Room
(E)xit
Which room do you want to go? >
```

選擇後，會將 passed 的變數設為 true ，結束遊戲。

Showing Status

此為顯示玩家的狀態，包括：玩家名稱、生命值、攻擊力、防禦力等，如下所示：

```
------------------ STATUS ------------------
userwei
> Health: 125/150
> Attack: 33
> Defense: 30
> Hunger: 3
> Thirst: 3
> Poison:
  - Poison_3
--------
> Items:
  - sword Wood Sword
--------
> Equipments:
  - Sword : Steel Sword
  - Shield: Wood Shield
--------
> Foods:
  - Water_1
  - Chocolate
--------------------------------------------
```

如果裝備是空的話則會顯示 (empty)，其餘則列舉於它們的分類中。
此功能是利用在 Player.cpp 中的 triggerEvent() 函數實現，此為 virtual function，會依傳入的 object 類型不同而決定輸出內容，在 player.cpp 中則是輸出玩家的狀態。

Pick up Items

在遇到 NPC 或在特定的房間中，可以選擇撿取物品，如下所示：

```
// Room
------------------- ROOM -------------------
> Room type: normal
> Hunger consumption: 0
> Thirst consumption: 0
0: NPC Guide Ko
1: sword Wood Sword
--------------------------------------------
Enter the instruction or the index of the charcter you want to interact: >

// NPC
I have these items. Do you want to get anything?
------------------- ITEM -------------------
0: food Water_1
1: food Chocolate
--------------------------------------------
Enter the index of wanted item (Enter -1 for quitting) >
```

玩家拾取的物品有兩種類型，分別是裝備與食物，可在選擇介面中裝備或使用。這裡先介紹裝備的部分，此時會呼叫 Player.cpp 中的 equip() 函式，玩家可以選擇自己要裝備的物品：

```
> Equipment:
  - Sword : (empty)
  - Shield: (empty)
(SW)ord
(SH)ield
(E)xit
Which equipment do you want to put on? >
```

接著會列出背包內的該類型裝備，並可以進行選擇所需的裝備：

```
Sword: (empty)
> Inventory:
  - 0: Wood Sword Attack: 1 Defense: 0
Enter the equipment you want to put on (Enter -1 to leave) >
```

選擇後即可看到目前的裝備：

```
> Equipment:
  - Sword : Wood Sword Attack: 1 Defense: 0
  - Shield: (empty)
(SW)ord
(SH)ield
(E)xit
Which equipment do you want to put on? >
```

各種裝備的數值範圍如下：

1. Wood Sword/Shield: 攻擊力/防禦力 1 至 10
2. Steel Sword/Shield: 攻擊力/防禦力 1 至 20
3. Titanium Sword/Shield: 攻擊力/防禦力 1 至 40
4. Diamond Sword/Shield: 攻擊力/防禦力 0 至 $2^{32}-1$（屬於通關獎勵，可在通關後炸魚）

若是選擇使用食物，則會列出背包內的食物，可供選擇使用：

```
------------------- FOOD -------------------
0: Water_1 usage: Restore 8 thirst
1: Chocolate usage: Restore 7 hunger
2: Cheese biscuit usage: Restore 5 hunger
3: Milk_2 usage: Remove 2 poison
4: Water_3 usage: Restore 7 thirst
--------------------------------------------
Which food you want to eat (-1 for quit) >
```

各種食物的數值範圍如下：

1. Water: 解除 1 至 10 點 thirst
2. Milk: 解除 1 至 3 種 poison
3. 其他食物: 解除 1 至 10 點 hunger

此時是直接利用 Dungeon.cpp 中的 chooseAction() 處理，並未額外呼叫其他函式。

Fighting System

此為戰鬥系統，是利用 Monster.cpp 中的 triggerEvent() 函數實現，在所在房間有怪物時會自動進入，如下圖所示：

```
Player userwei, you just encouter a monster.
Monster: Camel
Health: 30
Attack: 10
Defense: 5
(S)TATUS
(A)TTACK
(R)ETREAT
What do you want to do? >
```

玩家可以選擇攻擊或撤退，若選擇攻擊，則會進入攻擊模式，會使玩家與怪物互相攻擊，若怪物未被直接死亡則會對玩家造成傷害，如下圖所示：

```
Player userwei deals 16 damages to Monster Camel
Monster Camel deals 3 damages to Player userwei
(S)TATUS
(A)TTACK
(R)ETREAT
What do you want to do? >
```

若選擇撤退，則會回到上一個房間。

NPC

在遇到 NPC 時，會有一段對話，並可進行交互：

```
Welcome, player userwei!
I'm Guide Ko.
Here are some essentials to aid your journey.
Make sure to equip yourself before venturing deeper into the dungeon.

I have these items. Do you want to get anything?
------------------- ITEM -------------------
0: food Water_1
1: food Chocolate
--------------------------------------------
Enter the index of wanted item (Enter -1 for quitting) >
```

玩家可以選擇想要獲取的物品，如裝備、食物等，以降低在後續探索 Dungeon 的難度。
此功能是由 NPC.cpp 中的 triggerEvent() 函數實現，在進行交互後會將得到的物品移至背包中。另外 NPC 中所有的物件在初始化時會利用 mt19937 隨機指定該物件的數值，像是裝備的攻擊力、食物的恢復值等，詳細的數值範圍會在該項類別中說明。

Game Logic

這裡主要判斷兩個條件：玩家生命值是否為 0 或玩家是否到達出口。若是其中一個條件成立，則回傳 false，結束遊戲。若玩家生命值為 0，則會顯示以下畫面：

```
Player userwei is dead.
```

若玩家抵達出口，則會輸出：

```
Congratulations! Player userwei, you have exited the dungeon.
```

Hunger System Design

在此 Dungeon 中有加入 Hunger System 的功能，以下為其設計：

1. hunger
   hunger 會隨著玩家所在的房間不同有所變化，部分 NPC 會有食物可供玩家回復飢餓值。若 hunger 達到 0，則玩家會受到傷害，所受傷害為目前 hunger 與需消耗之差。

2. thirst
   thirst 會隨著玩家所在的房間不同有所變化，部分房間或 NPC 會提供水分可供玩家回復飢餓值。若飢餓值達到 0，則玩家會受到傷害，所受傷害為目前 thirstt 與需消耗之差。

3. poison
   poison 會在玩家抵達 swamp 房間時獲得，所獲得的 poison 值是隨機生成的，數值範圍為 1 至 5，並以 vector<Item> 的型態儲存，若未使用食物解除此負面效果則會導致玩家的生命值持續減少，減少的生命值為所有 poison 的數值之和。

這邊計算所有消耗、增加 poison、生命值折減的時機都在更換房間的時候，也就是若玩家一直進出 swamp 種類的房間，每次進入都會獲得一次 poison。

Room System Design

在 Room System 中，首先要設計遊戲地圖，並且在遊戲中進行移動時，會根據玩家的選擇進入不同的房間。以下是遊戲地圖的設計：

ExitRoom	forest3	desert4		
		normal1	swamp3	forest4
	normal3	desert3		desert2
	forest2		normal2	swamp2
	swamp1	desert1	forest1	
		StartRoom		

接下來會介紹所有種類房間的設計：

1. normal
   在 normal 的房間中，所受到的 hunger/thirst 消耗分別為 0/0，不會受到額外 poison 的攻擊，在此房間中會有機會遇到 NPC、隨機掉落的物品。

```
------------------- ROOM -------------------
> Room type: normal
> Hunger consumption: 0
> Thirst consumption: 0
0: NPC Guide Ko
1: sword Wood Sword
--------------------------------------------
```

2. forest
   在 forest 的房間中，所受到的 hunger/thirst 消耗分別為 4/1，不會受到額外 poison 的攻擊，在此房間中會遇到怪物、掉落的物品。其中有一間會有河流（以 NPC 形式出現），可提供水分給玩家補充 thirst。

```
------------------- ROOM -------------------
> Room type: forest
> Hunger consumption: 4
> Thirst consumption: 1
--------------------------------------------
```

3. desert
   在 desert 的房間中，所受到的 hunger/thirst 消耗分別為 1/4，不會受到額外 poison 的攻擊，在此房間中會遇到怪物、掉落的物品。其中有一間會有綠洲（以 NPC 形式出現），可提供水分給玩家補充 thirst。

```
------------------- ROOM -------------------
> Room type: desert
> Hunger consumption: 1
> Thirst consumption: 4
--------------------------------------------
```

4. swamp
   在 swamp 的房間中，所受到的 hunger/thirst 消耗分別為 2/2，進入後即會受到 poison 的攻擊，在此房間中會遇到怪物、掉落的物品。

```
------------------- ROOM -------------------
> Room type: swamp
> Hunger consumption: 2
> Thirst consumption: 2
> You are poisoned!
--------------------------------------------
```

5. exit
   在 exit 的房間中，玩家可以與 NPC 交互得到通關獎勵，並且可以選擇是否繼續遊戲或離開房間。

```
------------------- ROOM -------------------
> Room type: exit
> Hunger consumption: 0
> Thirst consumption: 0
0: NPC Guide Ko
--------------------------------------------
```

實作方式則是利用 Dungeon.cpp 中的 handleMovement() 及 Player.cpp 中的 debuffStates() 的函式，來進行相應的處理。

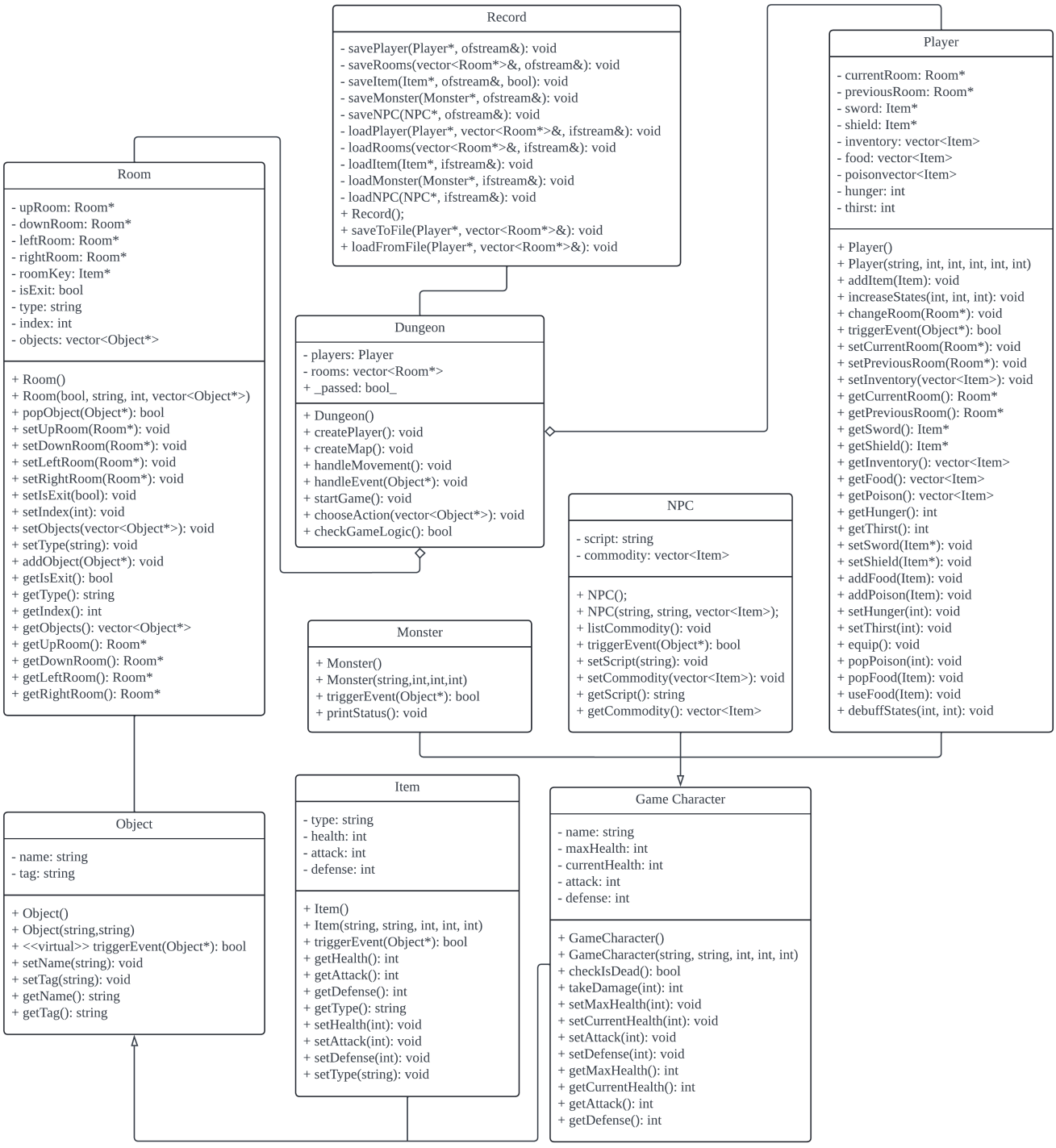
Optional Enhancement

這裡我選擇加上 Record System，可以將目前的遊戲進度儲存，並在下次遊戲時讀取，實作方式是利用 Record.cpp 中的 saveToFile() 和 loadFromFile() 函式，如下所示：

```cpp
void Record::saveToFile(Player* player, vector<Room*>& rooms){
    string filename = "record.txt";
    ofstream out;
    out.open(filename);
    saveRooms(rooms, out);
    savePlayer(player, out);
    out.close();
}

void Record::loadFromFile(Player* player, vector<Room*>& rooms){
    string filename = "record.txt";
    ifstream in;
    in.open(filename);
    if(in.fail()){
        cout << "Record file not found." << endl;
        return;
    }
    loadRooms(rooms, in);
    loadPlayer(player, rooms, in);
}
```

UML Design


Record
- savePlayer(Player*, ofstream&): void
- saveRooms(vector<Room*>&, ofstream&): void
- saveItem(Item*, ofstream&, bool): void
- saveMonster(Monster*, ofstream&): void
- saveNPC(NPC*, ofstream&): void
- loadPlayer(Player*, vector<Room*>&, ifstream&): void
- loadRooms(vector<Room*>&, ifstream&): void
- loadItem(Item*, ifstream&): void
- loadMonster(Monster*, ifstream&): void
- loadNPC(NPC*, ifstream&): void
+ Record();
+ saveToFile(Player*, vector<Room*>&): void
+ loadFromFile(Player*, vector<Room*>&): void
Player
- currentRoom: Room*
- previousRoom: Room*
- sword: Item*
- shield: Item*
- inventory: vector<Item>
- food: vector<Item>
- poisonvector<Item>
- hunger: int
- thirst: int
+ Player()
+ Player(string, int, int, int, int, int)
+ addItem(Item): void
+ increaseStates(int, int, int): void
+ changeRoom(Room*): void
+ triggerEvent(Object*): bool
+ setCurrentRoom(Room*): void
+ setPreviousRoom(Room*): void
+ setInventory(vector<Item>): void
+ getCurrentRoom(): Room*
+ getPreviousRoom(): Room*
+ getSword(): Item*
+ getShield(): Item*
+ getInventory(): vector<Item>
+ getFood(): vector<Item>
+ getPoison(): vector<Item>
+ getHunger(): int
+ getThirst(): int
+ setSword(Item*): void
+ setShield(Item*): void
+ addFood(Item): void
+ addPoison(Item): void
+ setHunger(int): void
+ setThirst(int): void
+ equip(): void
+ popPoison(int): void
+ popFood(Item): void
+ useFood(Item): void
+ debuffStates(int, int): void
Room
- upRoom: Room*
- downRoom: Room*
- leftRoom: Room*
- rightRoom: Room*
- roomKey: Item*
- isExit: bool
- type: string
- index: int
- objects: vector<Object*>
+ Room()
+ Room(bool, string, int, vector<Object*>)
+ popObject(Object*): bool
+ setUpRoom(Room*): void
+ setDownRoom(Room*): void
+ setLeftRoom(Room*): void
+ setRightRoom(Room*): void
+ setIsExit(bool): void
+ setIndex(int): void
+ setObjects(vector<Object*>): void
+ setType(string): void
+ addObject(Object*): void
+ getIsExit(): bool
+ getType(): string
+ getIndex(): int
+ getObjects(): vector<Object*>
+ getUpRoom(): Room*
+ getDownRoom(): Room*
+ getLeftRoom(): Room*
+ getRightRoom(): Room*
Dungeon
- players: Player
- rooms: vector<Room*>
+ _passed: bool_
+ Dungeon()
+ createPlayer(): void
+ createMap(): void
+ handleMovement(): void
+ handleEvent(Object*): void
+ startGame(): void
+ chooseAction(vector<Object*>): void
+ checkGameLogic(): bool
NPC
- script: string
- commodity: vector<Item>
+ NPC();
+ NPC(string, string, vector<Item>);
+ listCommodity(): void
+ triggerEvent(Object*): bool
+ setScript(string): void
+ setCommodity(vector<Item>): void
+ getScript(): string
+ getCommodity(): vector<Item>
Monster
+ Monster()
+ Monster(string,int,int,int)
+ triggerEvent(Object*): bool
+ printStatus(): void
Item
- type: string
- health: int
- attack: int
- defense: int
+ Item()
+ Item(string, string, int, int, int)
+ triggerEvent(Object*): bool
+ getHealth(): int
+ getAttack(): int
+ getDefense(): int
+ getType(): string
+ setHealth(int): void
+ setAttack(int): void
+ setDefense(int): void
+ setType(string): void
Game Character
- name: string
- maxHealth: int
- currentHealth: int
- attack: int
- defense: int
+ GameCharacter()
+ GameCharacter(string, string, int, int, int)
+ checkIsDead(): bool
+ takeDamage(int): int
+ setMaxHealth(int): void
+ setCurrentHealth(int): void
+ setAttack(int): void
+ setDefense(int): void
+ getMaxHealth(): int
+ getCurrentHealth(): int
+ getAttack(): int
+ getDefense(): int
Object
- name: string
- tag: string
+ Object()
+ Object(string,string)
+ <<virtual>> triggerEvent(Object*): bool
+ setName(string): void
+ setTag(string): void
+ getName(): string
+ getTag(): string

Results

將所有的程式碼上傳至 GitHub，並利用 CMake 產生 Release 檔一同發佈，網址如下：
https://github.com/chou-ting-wei/NYCU_OOP-Dungeon

Discussion

1. 如何在 Visual Studio Code 中完成 Dungeon 的開發？
   利用 CMake 來建立專案，並且在 CMakeLists.txt 中設定 includePath、compileCommands 等，使其可以建置專案。以下是 CMakeLists.txt 的內容：

```
cmake_minimum_required(VERSION 3.5.0)
project(dungeon)
message(STATUS "Project Directory: ${PROJECT_SOURCE_DIR}")

set(CMAKE_BUILD_TYPE Debug)
set(CMAKE_C_FLAGS_DEBUG "${CMAKE_C_FLAGS_DEBUG} -g -Wall -Wextra -Wshadow -
    Wconversion")
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

option(_LINUX "build the project on Linux" ON)
if(_LINUX)
    add_compile_definitions(_LINUX)
endif()

aux_source_directory(${CMAKE_SOURCE_DIR}/src DIR_SRC)
add_executable(dungeon ${DIR_SRC})
target_include_directories(dungeon PRIVATE ${CMAKE_SOURCE_DIR}/include)
```

2. 如何實作按下任意鍵以開始遊戲的功能？
   透過更改 ternimal 的設定，關閉 canonical mode 和 echo 的功能，使其達到按下任意鍵後開始遊戲的效果。以下是程式碼的部分：

```
#if defined(__unix__) || defined(__linux__) || defined(__APPLE__)
#include <unistd.h>
#include <termios.h>
#include <sys/ioctl.h>

inline void waitForKeypress() {
    struct termios oldt, newt;
    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~(ICANON | ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);
    cin.get();
    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
}
#endif
```

Conclusion

在這次的 Dungeon 作業中，我首次嘗試使用 C++ 來開發一個物件導向的遊戲。過去在高中時我有使用 Java 寫過圖形化的遊戲，但 C++ 在語法和輸出上有許多不同，讓我獲益良多。透過這次的專案，我深入了解了 C++ 中類別的使用方式，包括 Virtual Function、繼承類別、利用 Bash script 顯示顏色等，讓我對 OOP 有了更深的認識。雖然過程中遇到了許多挑戰，但這些都讓我學習到如何更有效地解決問題。總結來說，這是一次難得且有價值的學習經驗。